

# Fast Learning of Relational Dependency Networks

Oliver Schulte, Zhensong Qian, Arthur E. Kirkpatrick  
Xiaoqian Yin, Yan Sun

School of Computing Science, Simon Fraser University, Canada  
{oschulte,zqian,tet,xiaoqian\_yin,sunyans}@sfu.ca

**Abstract.** A Relational Dependency Network (RDN) is a directed graphical model widely used for multi-relational data. These networks allow cyclic dependencies, necessary to represent relational autocorrelations. We describe an approach for learning both the RDN’s structure and its parameters, given an input relational database: First learn a Bayesian network (BN), then transform the Bayesian network to an RDN. Thus fast Bayes net learning can provide fast RDN learning. The BN-to-RDN transform comprises a simple, local adjustment of the Bayes net structure and a closed-form transform of the Bayes net parameters. This method can learn an RDN for a dataset with a million tuples in minutes. We empirically compare our approach to state-of-the art RDN learning methods that use functional gradient boosting, on five benchmark datasets. Learning RDNs via BNs scales much better to large datasets than learning RDNs with boosting, and provides competitive accuracy in predictions.

## 1 Introduction

Learning graphical models is one of the main approaches to extending machine learning for relational data. Dependency networks (DNs) [5] are one of the major classes of graphical generative models, together with Markov networks and Bayesian networks (BNs) [14]. We describe a new approach to learning dependency networks: first learn a Bayesian network, then convert the Bayesian network to a dependency network. This hybrid approach combines the advantages of learning with Bayesian networks and performing inference with relational dependency networks. The hybrid learning algorithm produces dependency networks for large complex databases, with up to one million records, and up to 19 predicates. The predictive accuracy of the learned dependency networks is competitive with those constructed by state-of-the-art function gradient boosting methods. Bayesian network learning scales substantially better to larger datasets than the boosting methods. Our main contributions are:

1. A faster approach for learning relational dependency networks: first learn a Bayesian network, then convert it to a dependency network.
2. A closed-form log-linear discriminative model for computing the relational dependency network parameters from Bayesian network structure and parameters.

## 2 Relational Dependency Networks and Bayesian Networks

We review the definition of dependency networks and their advantages for modelling relational data. We assume familiarity with the basic concepts of Bayesian networks [14].

### 2.1 Dependency networks and Bayesian networks

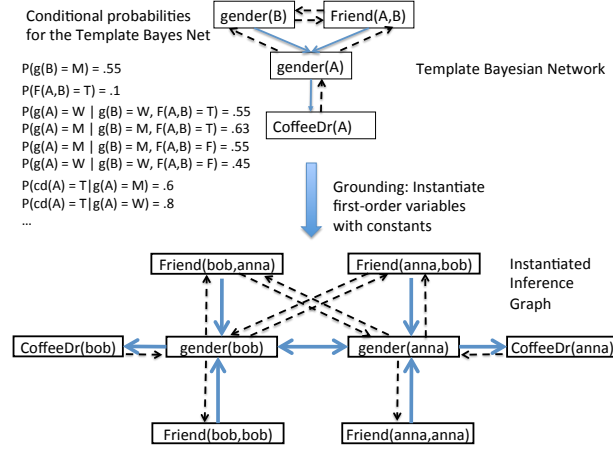
Like Bayesian networks, the structure of a dependency network is defined by a graph whose nodes are random variables and whose edges are directed. Unlike Bayesian networks, a dependency network graph may contain cycles and bi-directed edges. As with Bayesian networks, the parameters of dependency networks are conditional distributions over the value of a child node given its parents. The difference lies in the characteristic independence property of dependency networks: each node is independent of *all* other nodes given an assignment of values to its parents, which is generally not the case for Bayesian networks. In graphical model terms, the parents of a node in a dependency network form a Markov blanket: A minimal set of nodes such that assigning them values will make this node independent of the rest of the network.

Consequently, a parameter in a dependency network effectively specifies the probability of a node value given an assignment of values to all other nodes. We therefore refer to such conditional probabilities as **Gibbs conditional probabilities**, or simply Gibbs probabilities.<sup>1</sup> Gibbs sampling can be used to derive a joint distribution from the Gibbs probability DN parameters [5,13]. This is the counterpart to the Bayes net product formula that derives a joint distribution from the network’s conditional probability parameters.

### 2.2 Relational Dependency Networks

We use functor-based notation for graphical-relational models [15]. A functor is a function or predicate symbol. Each functor has a set of values (constants) called the **domain** of the functor. In this paper we consider only functors with finite domains. A **Parametrized Random Variable** (PRV) is of the form  $f(\tau_1, \dots, \tau_k)$  where  $f$  is a functor and each  $\tau_i$  is a first-order variable or a constant. A Parametrized Bayesian Network structure is a directed acyclic graph whose nodes are PRVs. A **relational dependency network structure** (RDN) is a directed graph whose nodes are PRVs. RDNs extend dependency networks for relational data by using knowledge-based model construction [13]: The first-order variables in a template RDN graph are instantiated for a specific domain of individuals to produce an *instantiated* or *ground* propositional DN graph, the **inference graph**. Figure 1 gives a dependency network template and its

<sup>1</sup> In the terminology of dependency networks [5], Gibbs probabilities are referred to as local probability distributions.



**Fig. 1.** A Bayesian/dependency template network (top) and the instantiated inference graphs (bottom). BN edges are shown as blue and solid. The BN-to-DN transformation adds the edges shown as black and dashed. Notice that grounding the BN induces a bi-directed edge between  $gender(bob)$  and  $gender(anna)$ .

grounded inference graph. An example Gibbs probability distribution for the inference graph (abbreviating functors to their first letter) is

$$P(g(anna) \mid g(bob), CD(anna), F(anna, bob), F(bob, anna), F(anna, anna)).$$

Both the structure and the parameter space of RDN models offer advantages for relational data [13,11]: (1) Dependency network structures are well-adapted for relational data because they allow cyclic dependencies, so grounding a dependency network template is guaranteed to produce a valid dependency network. (2) Relational prediction requires aggregating information from different linked individuals [12]. In a dependency network parameter, the aggregation encompasses the entire Markov blanket of a target node, whereas for Bayesian network parameters, the aggregation encompasses only its parents.

### 3 Learning Relational Dependency Networks via Bayesian Networks

Our algorithm for rapidly learning relational dependency networks begins with any relational learning algorithm for Bayesian networks. We then apply a simple, fast transformation of the resulting Bayesian network to a relational dependency template. Finally we apply a closed-form computation to derive the dependency network parameters from the Bayesian structure and parameters. Figure 2 shows the program flow.

Converting a Bayesian network structure to a dependency network structure is simple: for each node, add an edge pointing to the node from each member of its BN Markov blanket [5]. The result contains bidirectional links between each node, its children, and its co-parents (nodes that share a child with this one). This is equivalent to the standard moralization method for converting a BN to an undirected model [2], except that the dependency network contains bidirected edges instead of undirected edges. Bidirected edges have the advantage that they permit assignment of different parameters to each direction, whereas undirected edges have only one parameter.

Converting Bayesian network parameters to dependency network parameters is simple for propositional i.i.d. data: solve for the Gibbs conditional probabilities given Bayesian network parameters. The propositional result is as follows. A **family** comprises a node and its parents. A **family configuration** specifies a value for a child node and each of its parents. For example in the Bayesian network of Figure 1, a family configuration is

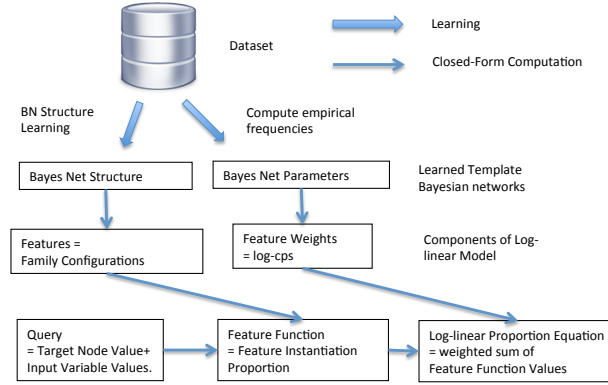
$$gender(\mathbb{A}) = M, Friend(\mathbb{A}, \mathbb{B}) = T, gender(\mathbb{B}) = M.$$

For propositional data, an assignment of values to the Markov blanket of a target node assigns a unique configuration for each family whose child is the target node or one of its children. Hence the Markov blanket induces a *unique* log-conditional probability for each such family configuration. The probability of a target node value given an assignment of values to the Markov blanket is then proportional to the exponentiated sum of these log-conditional probabilities [16, Ch.14.5.2].

With relational data, different family configurations such as the one above can be simultaneously instantiated, multiple times. We adapt the propositional log-linear equation for relational data by replacing the unique log-conditional probability with the *expected* log-conditional probability that results from selecting an instantiation of the family configuration uniformly at random. The probability of a target node value given an assignment of values to the Markov blanket is then proportional to the exponentiated sum of the expected log-conditional probabilities. We describe the resulting closed-form equation in the next section.

## 4 The Log-linear Proportion Equation

We propose a log-linear equation, the **log-linear proportion equation**, for computing a Gibbs conditional probability for a ground target node,  $T^*$ , given (i) a target value  $t$  for the target node, (ii) a complete set of values  $\Lambda^*$  for all ground terms other than the target node, and (iii) a template Bayesian network. The template structure is represented by functions that return the set of parent nodes of  $U$ ,  $\text{Pa}(U)$ , and the set of child nodes of  $U$ ,  $\text{Ch}(U)$ . The parameters of the template are represented by the conditional probabilities of a node  $U$  having a value  $u$  conditional on the values of its parents,  $\theta(U = u | \text{Pa}(U) = \mathbf{u}_{pa})$ . A grounding  $\gamma$  substitutes a constant for each member of a list of first-order variables. A grounding is therefore equivalent to an equality constraint  $\{\mathbb{A}_1 = a_1, \dots, \mathbb{A}_k = a_k\}$ . Applying a grounding to a template node defines a fully ground



**Fig. 2.** The program flow for computing Gibbs probabilities from a template Bayesian network. Features and weights are computed from the Bayes net. Feature function values are computed for each query.

target node. For instance, we may have  $gender(\mathbb{A})\{\mathbb{A} = sam\} = gender(sam)$ . These are combined in the following log-linear equation:

**Definition 1 (The Log-Linear Proportion Equation).**

$$P(T^* = t | \Lambda^*) \propto \sum_U \sum_{u, \mathbf{u}_{pa}} [\ln \theta(U = u | \text{Pa}(U) = \mathbf{u}_{pa})] \cdot p^r[\gamma; U = u, \text{Pa}(U) = \mathbf{u}_{pa}; T^* = t, \Lambda^*]$$

where

$U$  varies over  $\{T\} \cup \text{Ch}(T)$ ,

the singleton value  $u$  varies over the range of  $U$ ,

the vector of values  $\mathbf{u}_{pa}$  varies over the product of the ranges of  $U$ 's parents,

$T^* = T\gamma$  is the target node grounding of template node  $T$ , and  
 $p^r$  is the proportion feature function.

The feature function  $p^r$  specifies the proportion of instantiations that satisfy a given family configuration, relative to all family configurations with positive links only. This proportion is computed as follows.

1. For a given family configuration  $(U = u, \text{Pa}(U) = \mathbf{u}_{pa})$ , let the **family count**

$$n[\gamma; U = u, \text{Pa}(U) = \mathbf{u}_{pa}; T^* = t, \Lambda^*]$$

be the number of instantiations that (a) satisfy the family configuration and the ground node values specified by  $T^* = t, \Lambda^*$ , and (b) are consistent with

the equality constraint defined by  $\gamma$ . This notation is consistent with the parfactor notation of [15].

2. The **relevant family count**  $n^r$  is 0 if the family configuration contains a false relationship (other than the target node), else equals the feature count.
3. The **family proportion** is the relevant family count, divided by the total sum of all relevant family counts for the given family. In symbols:

$$p^r[\gamma; U = u, \text{Pa}(U) = \mathbf{u}_{pa}; T^* = t, \Lambda^*] = \frac{n^r[\gamma; U = u, \text{Pa}(U) = \mathbf{u}_{pa}; T^* = t, \Lambda^*]}{\sum_{u', \mathbf{u}'_{pa}} n^r[\gamma; U = u', \text{Pa}(U) = \mathbf{u}'_{pa}; T^* = t, \Lambda^*]}$$

It is common in statistical-relational models to restrict predictors to existing relationships only [3,16]. The inner sum of Formula 1 computes the expected log-conditional probability for a family with child node  $U$ , when we randomly select a relevant grounding of the first-order variables in the family.

Definition 1 has the form of a log-linear model [20]: The features of the model are the family configurations  $(U = u, \text{Pa}(U) = \mathbf{u}_{pa})$  where the child node is either the target node or one of its children. The feature weights are the log-conditional BN probabilities defined for the family configuration. The input variables are the values of the ground nodes other than the target nodes, specified by the conjunction  $\Lambda^*$ . The family count specifies how many times the feature is instantiated in the input variables (plus the target node value). The family proportion is the feature function, which maps a feature to a real value given the input variables. Proportions have the desirable consequence that all feature functions are normalized to the  $[0,1]$  range.

*Example.* Table 1 illustrates the computation of our log-linear model for predicting the gender of a new test instance (*sam*).

*Estimating Bayes net parameters.* The Bayesian network parameters can be estimated by applying the maximum likelihood principle, which entails using the empirical conditional frequencies observed in an input relational database [17,19]. Although there is theoretical justification for using the empirical frequencies, the ultimate test is whether the method can achieve comparable accuracy and greater speed than prior methods of computing relational dependency networks. In the next section, we empirically compare these methods.

## 5 Empirical Comparison with Functional Gradient Boosting

The next section describes experiments that compare learning RDNs via Bayesian networks with functional gradient methods for learning relational dependency networks. Boosting methods follow the traditional approach to learning dependency networks, which is to learn a collection of separate discriminative models, one for each node in the network [5]. Functional gradient boosting has been shown to perform well on small datasets previously [8,11]; our experiments provide new new tests of this method on medium to large datasets.

**Table 1.** Applying the log-linear proportion equation with the Bayesian network of Figure 1 to compute  $P(\text{gender}(\text{sam}) = W|\Lambda^*)$  and  $P(\text{gender}(\text{sam}) = M|\Lambda^*)$ . Each row represents a feature/family configuration. For the sake of the example we suppose that the conjunction  $\Lambda^*$  specifies that Sam is a coffee drinker, has 60 male friends, and 40 female friends.  $CP$  refers to the conditional probability BN parameter of Figure 1. For the feature weights  $w \equiv \ln(CP)$ .

Child Value $u$	Parent State $\mathbf{u}_{pa}$	CP	$w$	$p^r$	$w \times p^r$
$g(\text{sam}) = W$	$g(B) = W,$ $F(\text{sam}, B) = T$	0.55	-0.60	0.4	-0.24
$g(\text{sam}) = W$	$g(B) = M,$ $F(\text{sam}, B) = T$	0.37	-0.99	0.6	-0.60
$CD(\text{sam}) = T$	$g(\text{sam}) = W$	0.80	-0.22	1.0	-0.22
$CD(\text{sam}) = F$	$g(\text{sam}) = W$	0.20	-1.61	0.0	0.00
Sum ( $\exp(\text{Sum}) \propto P(\text{gender}(\text{sam}) = W \Lambda^*)$ )					-1.06
$g(\text{sam}) = M$	$g(B) = W,$ $F(\text{sam}, B) = T$	0.45	-0.80	0.4	-0.32
$g(\text{sam}) = M$	$g(B) = M,$ $F(\text{sam}, B) = T$	0.63	-0.46	0.6	-0.28
$CD(\text{sam}) = T$	$g(\text{sam}) = M$	0.60	-0.51	1.0	-0.51
$CD(\text{sam}) = F$	$g(\text{sam}) = M$	0.40	-0.92	0.0	0.00
Sum ( $\exp(\text{Sum}) \propto P(\text{gender}(\text{sam}) = M \Lambda^*)$ )					-1.11

### 5.1 Experimental Conditions and Metrics

All experiments were done on with 8GB of RAM and a single Intel Core 2 QUAD Processor Q6700 with a clock speed of 2.66GHz (there is no hyper-threading on this chip). The operating system was Linux Centos 2.6.32. Code was written in Java, JRE 1.7.0. All code and datasets are available [7].

**Datasets** We used 5 benchmark real-world databases. For more details please see the references in [18]. Summary statistics appear in Table 2.

**MovieLens Databases** MovieLens is a commonly-used rating dataset<sup>2</sup>. We added more related attribute information about the actors, directors and movies from the Internet Movie Database (IMDB)<sup>3</sup>. It contains two entity sets, Users and Movies. For each user and movie that appears in the database, all available ratings are included. MovieLens(1M) contains 1M ratings, 3,883 Movies, and 6,039 Users. MovieLens(0.1M) contains about 0.1M ratings, 1,682 Movies, and 941 Users. We did not use the binary genre predicates because they are easily learned with exclusion rules.

**Mutagenesis Database** This dataset is widely used in Inductive Logic Programming research. It contains information on Atoms, Molecules, and Bonds between them. We use the discretization of [18].

<sup>2</sup> [www.grouplens.org](http://www.grouplens.org)

<sup>3</sup> [www.imdb.com](http://www.imdb.com), July 2013

**Hepatitis Database** This data is a modified version of the PKDD02 Discovery Challenge database. The database contains information on the laboratory examinations of hepatitis B and C infected patients.

**Mondial Database** This dataset contains data from multiple geographical web data sources.

**UW-CSE Database** This dataset lists facts about the Department of Computer Science and Engineering at the University of Washington, such as entities (e.g., *Person*, *Course*) and the relationships (i.e. *AdvisedBy*, *TaughtBy*).

**Methods Compared** Functional gradient boosting is a state-of-the-art method for applying discriminative learning to build a generative graphical model. The local discriminative models are ensembles of relational regression trees [8]. Our experiments used the BoostR implementation of relational gradient boosting [9]. The current implementation does not support multi-class boosting, so following previous experiments [8], we limited our comparison to *binary predicates*, i.e., functors that can take on only two possible values (e.g., *AdvisedBy*). We compared the following three learning methods.

**RDN\_Bayes** Learns a Bayesian network, then converts it to a relational dependency network as described above.

**RDN\_Boost** The state-of-the-art gradient boosting method designed for learning RDNs. Information from ground nodes that are linked to the target node is aggregated with functions *count*, *max*, *average* and existential quantification [11].

**MLN\_Boost** The state-of-the-art gradient boosting method designed for learning Markov Logic Networks. It takes as input a list of target predicates for analysis. To construct an RDN, we provide each binary predicate as a single target predicate in turn. Information from ground nodes that are linked to the target node is aggregated with a log-linear model derived from Markov Logic Networks.

We followed the BoostR instructions for creating the background .bk file and used the default settings. We experimented with alternative settings but they did not improve the performance of the boosting methods.

To obtain the BN structure for RDN\_Bayes, the learn-and-join algorithm [18] was applied to each benchmark database. The BN parameters were computed from the empirical conditional frequencies in the database using previously-published algorithms [19].

**Prediction Metrics** We follow [8] and evaluate the algorithms using conditional log likelihood (CLL) and AUC-PR (Area Under Precision-Recall Curve). AUC-PR is appropriate when the target predicates features a skewed distribution as is typically the case with relationship predicates. For each fact  $T^* = t$  in the test dataset, we evaluate the accuracy of the predicted Gibbs probability  $P(T^* = t | \Lambda^*)$ , where  $\Lambda^*$  is a complete conjunction for all ground terms other



than  $T^*$ . Thus  $A^*$  represents the values of the input variables as specified by the test dataset. CLL is the average of the logarithm of the Gibbs probability for each ground truth fact in the test dataset. For the gradient boosting method, we used the AUC-PR and likelihood scoring routines included in BoostR.

Both metrics are reported as averages over all binary predicates. The learning methods were evaluated using 5-fold cross-validation. Each database was split into 5 folds by randomly selecting entities from each entity table, and restricting the relationship tuples in each fold to those involving only the selected entities (i.e., subgraph sampling [18]). The models were trained on 4 of the 5 folds, then tested on the remaining one. All results are averages from 5-fold cross validation, over all descriptive attributes in the database.

## 5.2 Results

Table 2 shows learning times for the different methods. For the boosting method, we added together the learning times for each target predicate. The total learning times are not directly comparable because Bayes net learning simultaneously learns a joint model for all predicates. We therefore report total learning time divided by the number of all predicates for RDN\_Bayes, and total learning time divided by the number of binary predicates for the boosting methods. The numbers of predicates are given in the second column.

**Table 2.** Learning Time (Sec) Per Predicate

Dataset	all predicates / binary pred- icates	# tuples	RDN_Bayes	RDN_Boost	MLN_Boost
UW	14/4	612	0.74±0.05	14.57±0.39	19.27±0.77
Mondial	18/4	870	101.53±6.90	27.21±0.98	41.97±1.03
Hepatitis	19/7	11,316	285.71±20.94	250.61±5.32	229.73±2.04
Mutagenesis	11/6	24,326	0.70±0.02	117.70±6.37	48.65±1.34
MovieLens(0.1M)	7/2	83,402	1.11±0.08	2638.71±272.78	1866.605±112.54
MovieLens(1M)	7/2	1,010,051	1.12±0.10	>24 hours	>24 hours

Table 2 shows that RDN\_Bayes scales very well with the number of data tuples: even the large MovieLens dataset with 1M records can be analyzed in seconds. Learning separate discriminative models scales well with the number of predicates, which is consistent with findings from propositional learning [5]. Bayes net learning slows down more as more predicates are included, since it learns a joint model over all predicates simultaneously. However, the learning time remains feasible (see also [18]). Bayesian network learning scales well in the number of data points because it provides closed-form parameter estimation and hence closed-form model scoring. Unlike propositional iid data, relational data are represented in multiple tables, so model evaluation requires expensive combining of information from different tables [13]. Compared to learning separate

discriminative models, Bayesian network explores a more complex model space, but model evaluation is much faster.

**Table 3.** Average Conditional Log-Likelihood

CLL	UW	Mondial	Hepatitis	Mutagenesis	MovieLens(0.1M)
RDN_Boost	-0.29±0.02	-0.48±0.03	-0.51±0.00	-0.43±0.02	-0.58±0.05
MLN_Boost	-0.16±0.01	-0.40±0.05	-0.52±0.00	-0.24±0.02	-0.38±0.06
RDN_Bayes	<b>-0.01±0.00</b>	<b>-0.25±0.06</b>	<b>-0.39±0.10</b>	<b>-0.22±0.07</b>	<b>-0.30±0.02</b>

**Table 4.** Average Area Under Precision-Recall Curve

AUC-PR	UW	Mondial	Hepatitis	Mutagenesis	MovieLens(0.1M)
RDN_Boost	0.32±0.01	0.27±0.01	<b>0.71±0.02</b>	0.63±0.02	0.52±0.03
MLN_Boost	0.52±0.01	0.44±0.05	<b>0.71±0.02</b>	<b>0.83±0.05</b>	0.52±0.05
RDN_Bayes	<b>0.89±0.00</b>	<b>0.79±0.07</b>	0.55±0.11	0.50±0.10	<b>0.65±0.02</b>

Tables 3 and 4 show results for predictive accuracy. Our system resources did not suffice for evaluating the metrics on MovieLens(1M). In terms of the likelihood assigned to the ground truth predicate value, the Bayes net method outperforms both boosting methods on all datasets (Table 3). In terms of the precision-recall curve, the Bayes net method performs substantially better than both on three datasets, and substantially worse on the two others (Table 4). This is a satisfactory performance because boosting is a powerful method for achieving accurate predictions, and was applied to each target predicate individually to produce a tailored discriminative model. Bayesian network learning simultaneously constructed a joint model for all predicates, and used simple maximum likelihood estimation for parameter values. Our overall conclusion is that *Bayes net learning scales much better to large datasets, and provides competitive accuracy in predictions.*

In addition to scalability, Bayesian networks offer two more advantages. First, learning easily extends to attributes with more than two possible values. Second, the parameters and the predictions derived from them are easily interpretable. The ensemble of regression trees is more difficult to interpret, as the inventors of the boosting method noted [11].

## 6 Related Work

Dependency networks were introduced in [5] and relational dependency networks in [13]. Heckerman *et al.* compare Bayesian, Markov and dependency networks for nonrelational data.

*Bayesian networks.* There are several proposals for defining directed relational template models, based on graphs with directed edges or rules in clausal format [6,3]. Defining the probability of a child node conditional on multiple instantiations of a parent set requires the addition of combining rules [6] or aggregation functions [3]. Combining rules such as the arithmetic mean [12] combine

global parameters with a local scaling factor, as does our log-linear model. In terms of combining rules, our model uses the *geometric mean* rather than the arithmetic mean.<sup>4</sup> To our knowledge, the geometric mean has not been used before as a combining rule for relational data.

*Markov Networks.* Markov Logic Networks (MLNs) provide a logical template language for undirected graphical models. Richardson and Domingos propose transforming a Bayesian network to a Markov Logic network using moralization, with log-conditional probabilities as weights [2]. This is also the standard BN-to-MLN transformation recommended by the Alchemy system [1]. A discriminative model can be derived from any MLN [2]. The structure transformation was used in previous work [18], where MLN parameters were learned, not computed in closed-form from BN parameters. The Gibbs conditional probabilities derived from an MLN obtained from converting a Bayesian network are the same as those defined by our log-linear Formula 1, *if* counts replace proportions as feature functions [17]. There is no MLN whose discriminative model is equivalent to our log-linear equation with proportions as feature functions.<sup>5</sup>

## 7 Conclusion and Future Work

Relational dependency networks offer important advantages for modelling relational data. We proposed a novel approach to learning dependency networks: first learn a Bayesian network, then perform a closed-form transformation of the Bayesian network to a dependency network. The key question is how to transform BN parameters to DN parameters. We introduced a new relational adaptation of the standard BN log-linear equation for the probability of a target node conditional on an assignment of values to its Markov blanket. The new log-linear equation uses a sum of expected values of BN log-conditional probabilities, with respect to a random instantiation of first-order variables. This is equivalent to using feature instantiation proportions as feature functions. We compared our approach to state-of-the-art functional gradient boosting methods on five benchmark datasets. Learning RDNs via BNs scales much better to large datasets than with boosting, and provides competitive accuracy in predictions.

Learning a collection of discriminative models and learning a Bayesian network learning are two very different approaches to constructing dependency networks, each with strengths and weaknesses. There are various options for hybrid approaches that combine the strengths of both. (1) Fast Bayesian network learning methods can be used to select features. Discriminative learning methods should work faster restricted to the BN Markov blanket of a target node. (2) The Bayesian network can provide an initial dependency network structure.

<sup>4</sup> The geometric mean of a list of numbers  $x_1, \dots, x_n$  is  $(\prod_i x_i)^{1/n}$ . The logarithm of the geometric mean is therefore  $1/n \sum_i \ln x_i$ . Thus geometric mean =  $\exp(\text{average}(\text{logs}))$ .

<sup>5</sup> Disclaimer: A preliminary version of this paper was presented at the StarAI 2012 workshop, with no archival proceedings. We are indebted to workshop reviewers and participants for helpful comments.

Gradient boosting can then be used to fine-tune a discriminative model of a child node given parent nodes, replacing a flat conditional probability table. In sum, learning relational dependency networks via Bayesian networks is a novel approach that offers promising advantages for interpretability and scalability.

## Acknowledgements

This work was supported by Discovery Grants to Oliver Schulte from the Natural Science and Engineering Council of Canada. Zhensong Qian was supported by a grant from the China Scholarship Council.

## Appendix: Proof of Consistency Characterization

We show that for a given template BN, there are two ground target nodes and query conjunction  $\Lambda^*$  such that the conditional distributions of the ground target nodes given  $\Lambda^*$  do not agree with any joint distribution over the ground target nodes given  $\Lambda^*$ . We begin by establishing some properties of the template BN and the query conjunction that are needed in the second part of the proof. The second part proves the inconsistency by showing that consistency entails a constraint that is violated by the template BN for the constructed query conjunction  $\Lambda^*$ .

### 7.1 Properties of the template BN and the input query $\Lambda^*$

The inconsistency of the BN networks arises when a parent and a child ground node have different relevant family counts. The next lemma shows that this is possible exactly when the template BN is properly relational, meaning it relates parents and children from different populations.

**Lemma 1.** *The following conditions are equivalent for a template edge  $T_1 \rightarrow T_2$ .*

1. *The parent and child do not contain the same population variables.*
2. *It is possible to find a grounding  $\gamma$  for both parent and child, and an assignment  $\Lambda^*$  to all other nodes, such that the relevant family count for the  $T_2$  family differs for  $T_1^* = \gamma T_1$  and  $T_2^* = \gamma T_2$ .*

*Proof.* If the parent and child contain the same population variables, then there is a 1-1 correspondence between groundings of the child and groundings of the parents. Hence the count of relevant family groundings is the same for each, no matter how parents and child are instantiated. If the parent and child do not contain the same population variables, suppose without loss of generality that the child contains a population variable  $\mathbb{A}$  not contained in the parent. Choose a common grounding  $\gamma$  for the parents and child node. For the ground child node,  $\gamma T_2$ , let  $\gamma$  be the only family grounding that is relevant, so the relevant count is 1. For the ground parent node, there is at least one other grounding of the child node  $T_2'$  different from  $\gamma T_2$  since  $T_2$  contains another population variables. Thus it is possible to add another relevant family grounding for  $\gamma T_1$ , which means that the relevant count is at least 2.

The proof proceeds in the most simple manner if we focus on template edges that different populations and have no common children.

**Definition 2.** *An template edge  $T_1 \rightarrow T_2$  is **suitable** if*

1. *The parent and child do not contain the same population variables.*
2. *The parent and child have no common edge.*

The next lemma shows that focusing on suitable edges incurs no loss of generality.

**Lemma 2.** *Suppose that a template BN contains an edge such that the parent and child do not contain the same population variables. Then the template BN contains a suitable edge.*

*Proof.* Suppose that there is an edge satisfying the population variable condition. Suppose that the parent and child share a common child. Since the edge satisfies the condition, the set of population variables in the common child differs from at least one of  $T_1, T_2$ . Therefore there is another edge from one of  $T_1 \rightarrow T_2$  as parent to a new child that satisfies the population variable condition. If this edge is not suitable, there must be another shared child. Repeating this argument, we eventually arrive at an edge satisfying the population variable condition where the child node is a sink node without children. This edge is suitable.

Consider a suitable template edge  $T_1 \rightarrow T_2$  that produces a bidirected ground edge  $T_1^* \leftrightarrow T_2^*$ . For simplicity we assume that  $T_1$  and  $T_2$  are binary variables with domain  $\{T, F\}$ . (This incurs no loss of generality as we can choose a database  $A^*$  in which only two values occur.) Let  $\text{Pa}(T_2)$  be the parents of  $T_2$  other than  $T_1$ . Since the template edge is not redundant [14], there is a parent value setting  $\text{Pa}(T_2) = \mathbf{pa}$  such that  $T_1$  and  $T_2$  are conditionally dependent given  $\text{Pa}(T_2) = \mathbf{pa}$ . This implies that the conditional distribution of  $T_1$  is different for each of the two possible values of  $T_2$ : In terms of the template Bayesian network parameters, this implies that

$$\frac{\theta(T_2 = F | T_1 = F, \mathbf{pa})}{\theta(T_2 = T | T_1 = F, \mathbf{pa})} \neq \frac{\theta(T_2 = F | T_1 = T, \mathbf{pa})}{\theta(T_2 = T | T_1 = T, \mathbf{pa})}. \quad (1)$$

Let  $A^*$  denote an assignment of values to all ground nodes other than the target nodes  $T_1^*$  and  $T_2^*$ . We assume that the input query  $A^*$  assigns different relevant family counts  $N_1$  to  $T_1^*$  and  $N_2$  to  $T_2^*$ . This is possible according to Lemma 1.

## 7.2 Lowd's Equation and Relevant Family Counts

The log-linear equation 1, specifies the conditional distribution of each target node given  $A^*$  and a value for the other target node. We keep the assignment  $A^*$  fixed throughout, so for more compact notation, we abbreviate the conditional distributions as

$$p(T_1^* = t_1 | T_2^* = t_2) \equiv P(T_1^* = t_1 | T_2^* = t_2, \Lambda^*)$$

and similarly for  $P(T_1^* = t_1 | T_2^* = t_2, \Lambda^*)$ .

On the assumption that the dependency network is consistent, there is a joint distribution over the target nodes conditional on the assignment that agrees with the conditional distribution:

$$\frac{p(T_1^* = t_1, T_2^* = t_2)}{p(T_2^* = t_2)} = p(T_1^* = t_1 | T_2^*)^*$$

and also with the conditional  $p(T_2^* = t_2 | T_1^* = t_1)$ .

Lowd [10] pointed out that this joint distribution satisfies the equations

$$\frac{p(F, F)}{p(T, F)} \cdot \frac{p(T, F)}{p(T, T)} = \frac{p(F, F)}{p(T, T)} = \frac{p(F, F)}{p(F, T)} \cdot \frac{p(F, T)}{p(T, T)} \quad (2)$$

Since the ratio of joint probabilities is the same as the ratio of conditional probabilities for the same conditioning event, consistency entails the following constraint on conditional probabilities via Equation (2):

$$\frac{p(T_2^* = F | T_1^* = F)}{p(T_2^* = T | T_1^* = F)} \cdot \frac{p(T_1^* = F | T_2^* = T)}{p(T_1^* = T | T_2^* = T)} = \frac{p(T_1^* = F | T_2^* = F)}{p(T_1^* = T | T_2^* = F)} \cdot \frac{p(T_2^* = F | T_1^* = T)}{p(T_2^* = T | T_1^* = T)} \quad (3)$$

We refer to Equation 3 as *Lowd's equation*. The idea of our proof is to show that Lowd's equations are satisfied only if the relevant family counts for the target nodes are the same. According to the log-linear equation, each conditional probability is proportional to a product of BN parameters. The first step is to show that in Lowd's equation, all BN parameter terms cancel out except for those that are derived from the family that comprises  $T_1^*$  and their  $T_2^*$  and their common grounding. This may not hold in general, but can be proved provided that the edge  $T_1 \rightarrow T_2$  satisfies two conditions.

**Definition 3.** *An template edge  $T_1 \rightarrow T_2$  is **suitable** if*

1. *It is possible to find a grounding  $\gamma$  for both parent and child, and an assignment  $\Lambda^*$  to all other nodes, such that the relevant family count for the  $T_2$  family differs for  $T_1^* = \gamma T_1$  and  $T_2^* = \gamma T_2$ .*
2. *the parent and child have no common edge.*

**Lemma 3.** *The conditional probabilities for the target nodes can be written as follows:*

$$P(T_2^* = t_2 | T_1^* = t_1, \Lambda^*) \propto \theta(T_2 = t_2 | T_1 = t_1, \mathbf{pa})^{(N/N_2 + M_{T_2=t_2}/N_2)} \cdot \pi_{T_2=t_2} \quad (4)$$

where  $M_{T_2=t_2}$  and  $\pi_{T_2=t_2}$  depend only on  $t_2$  and not on  $t_1$  and

$$P(T_1^* = t_1 | T_2^* = t_2, \Lambda^*) \propto \theta(T_2 = t_2 | T_1 = t_1, \mathbf{pa})^{(N/N_1 + M_{T_1=t_1}/N_1)} \cdot \pi_{T_1=t_1} \quad (5)$$

where  $M_{T_1=t_1}$  and  $\pi_{T_1=t_1}$  depend only on  $t_1$  and not on  $t_2$ .

*Proof.* We start with target node  $T_2^*$ . (1) The log-linear equation 1 contains a term for the children of  $T_2^*$ . Since  $T_1$  and  $T_2$  share no children, the corresponding conditional probabilities do not depend on the value of  $T_1^*$ , but only on  $\Lambda^*$  and  $T_2$ . Thus the product of the BN parameters can be denoted as  $\pi_{T_2=t_2}$ . (2) The only other term in the log-linear equation is for the family of  $T_2^*$ . Since  $\Lambda^*$  is suitable, the only instantiated groundings for the parents of  $T_2^*$  agree with the values  $\mathbf{pa}$ . These groundings can be divided into those that agree with  $T_1^*$  and those that do not. (3) The log-linear terms for the latter do not depend on the value  $t_1$  of  $T_1^*$ , hence their number can be written as  $M_{T_2=t_2}$ . (4) For groundings that are consistent with both  $T_1^*$  and  $T_2^*$ , their number does not depend on the values of  $T_1^*$  or  $T_2^*$ . It depends only on  $\Lambda^*$ . Let this number be  $N$ .

Now consider target node  $T_1^*$ . (1) The log-linear equation 1 contains a term for the family of  $T_1^*$ . Since  $T_1$  is a parent of  $T_2$ , the acyclicity of the template BN entails that  $T_2$  is not a parent of  $T_1$ . Therefore the conditional probabilities for the family of  $T_1^*$  do not depend on the value of  $T_2^*$ , but only on  $\Lambda^*$  and  $T_1$ . (2) The log-linear equation 1 also contains a term for the children of  $T_1$  other than  $T_2$ . Since the edge  $T_1 \rightarrow T_2$  is suitable, the two nodes do not share a child, so these terms also do not depend on the value of  $T_2^*$ . Thus collectively, the product of the terms (1) and (2) can be written as  $\pi_{T_1=t_1}$ . The remaining terms are groundings for  $T_1^*$  and the family of  $T_2$ . These groundings can be divided into those that agree with  $T_2^*$  and those that do not. (3) The log-linear terms for the latter do not depend on the value  $t_2$  of  $T_2^*$ , hence their number can be written as  $M_{T_1=t_1}$ . (4) The number of groundings that are consistent with both  $T_1^*$  and  $T_2^*$  is denoted by  $N$  as above.

**Lemma 4.** *Suppose that conditions (4) and (5) of Lemma 3 hold. Then Lowd's Equation (3) holds if and only if  $N_1 = N_2$ .*

*Proof.* Observe that in Equation (3), each term on the left has a corresponding term with the same value for the target node assignment and the opposing conditioning assignment. For instance, the term  $p(T_2^* = F | T_1^* = F)$  on the left is matched with the term  $p(T_2^* = F | T_1^* = T)$  on the right. This means that the products in the log-linear expression are the same on both sides of the equation except for those factors that depend on *both*  $t_1$  and  $t_2$ . Continuing the example, the factors

$$\theta(T_2 = F | T_1 = F, \mathbf{pa})^{(M_F/N_2)} \cdot \pi_{T_2=t_2}$$

on the left equal the factors

$$\theta(T_2 = F | T_1 = T, \mathbf{pa})^{(M_{T_1=t_1}/N_2)} \cdot \pi_{T_2=t_2}$$

on the right side of the equation. They therefore cancel out, leaving only the term

$$\theta(T_2 = F | T_1 = F, \mathbf{pa})^{N/N_2}$$

on the left and the term

$$\theta(T_2 = F | T_1 = F, \mathbf{pa})^{N/N_2}$$

on the right. Lowd’s equation can therefore be reduced to an equivalent constraint with only such BN parameter terms. For further compactness we abbreviate such terms as follows

$$\theta(t_2|t_1) \equiv \theta(T_2 = t_2|T_1 = t_1, \mathbf{pa}).$$

With this abbreviation, the conditions of Lemma 3 entail that Lowd’s equation 3 reduces to the equivalent expressions.

$$\frac{\theta(F|F)^{N/N_2}}{\theta(T|F)^{N/N_2}} \cdot \frac{\theta(T|F)^{N/N_1}}{\theta(T|T)^{N/N_1}} = \frac{\theta(F|F)^{N/N_1}}{\theta(F|T)^{N/N_1}} \cdot \frac{\theta(F|T)^{N/N_2}}{\theta(T|T)^{N/N_2}} \quad (6)$$

$$\left(\frac{\theta(F|F)}{\theta(T|F)}\right)^{(N/N_2 - N/N_1)} = \left(\frac{\theta(F|T)}{\theta(T|T)}\right)^{(N/N_2 - N/N_1)} \quad (7)$$

By the nonredundancy assumption (1) on the BN parameters, we have

$$\frac{\theta(F|F)}{\theta(T|F)} \neq \frac{\theta(F|T)}{\theta(T|T)}$$

so Equation 7 implies that

$$N_1 = N_2,$$

which establishes the lemma.

The main theorem now follows as follows: Lemma 1 entails that if the dependency network is consistent, the log-linear equations satisfy Lowd’s equation with the bidirected ground edge  $T_1^* \leftrightarrow T_2^*$  and the query conjunction  $\Lambda^*$  that satisfies the BN non-redundancy condition. Lemmas 3 and 2 show that if the template BN is relational, it must contain a suitable edge  $T_1 \rightarrow T_2$ . Lemma 4 together with Lowd’s equation entails that the relevant counts for  $T_1^*$  and  $T_2^*$  must then be the same. But the query conjunction  $\Lambda^*$  was chosen so that the relevant counts are different. This contradiction shows that Lowd’s equation is unsatisfiable, and therefore no joint distribution exists that is consistent with the BN conditional distributions specified by the log-linear Equation 1.

## References

1. Alchemy Group. Frequently asked questions. URL = <http://alchemy.cs.washington.edu/>.
2. Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan and Claypool Publishers, 2009.
3. Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. In *Introduction to Statistical Relational Learning* [4], chapter 5, pages 129–173.
4. Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. MIT Press, 2007.



5. David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, Carl Kadie, and Pack Kaelbling. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.
6. Kristian Kersting and Luc de Raedt. Bayesian logic programming: Theory and tool. In *Introduction to Statistical Relational Learning* [4], chapter 10, pages 291–318.
7. H. Khosravi, T. Man, J. Hu, E. Gao, and O. Schulte. Learn and join algorithm code. URL = <http://www.cs.sfu.ca/~oschulte/jbn/>.
8. Tushar Khot, Sriraam Natarajan, Kristian Kersting, and Jude W. Shavlik. Learning Markov logic networks via functional gradient boosting. In *ICDM*, pages 320–329. IEEE Computer Society, 2011.
9. Tushar Khot, Jude Shavlik, and Sriraam Natarajan. BoostR, 2013. URL = <http://pages.cs.wisc.edu/~tushar/BoostR/>.
10. D. Lowd. Closed-form learning of Markov networks from dependency networks. In *UAI*, 2012.
11. Sriraam Natarajan, Tushar Khot, Kristian Kersting, Bernd Gutmann, and Jude W. Shavlik. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning*, 86(1):25–56, 2012.
12. Sriraam Natarajan, Prasad Tadepalli, Thomas G. Dietterich, and Alan Fern. Learning first-order probabilistic models with combining rules. *Annals of Mathematics and Artificial Intelligence*, 54(1-3):223–256, 2008.
13. Jennifer Neville and David Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, 2007.
14. J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
15. David Poole. First-order probabilistic inference. In *IJCAI*, pages 985–991, 2003.
16. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.
17. Oliver Schulte. A tractable pseudo-likelihood function for Bayes nets applied to relational data. In *SIAM SDM*, pages 462–473, 2011.
18. Oliver Schulte and Hassan Khosravi. Learning graphical models for relational data via lattice search. *Machine Learning*, 88(3):331–368, 2012.
19. Oliver Schulte, Hassan Khosravi, Arthur Kirkpatrick, Tianxiang Gao, and Yuke Zhu. Modelling relational statistics with bayes nets. *Machine Learning*, 94:105–125, 2014.
20. Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning* [4], chapter 4, pages 93–127.